3/9/2004

PORTAL
THE ACM DIGITAL LIBRARY

> home > about > feedback > login

US Patent & Trademark Office

Beta Try the *new* Portal design
Give us your opinion after using it.

Search Results

Search Results for: [(risc and cisc)<AND>((((dual or two) <near> (operating <near> system)) <near> (exception or trap)))]
Found 196 of 127,944 searched.

Search within Results

GO > Advanced Search

> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 20 of 196 short listing

Prev Page 1 2 3 4 5 6 7 8 9 10 Next Page

1 RISCs vs. CISCs for Prolog: a case study 94%
Gaetano Borriello , Andrew R. Cherenson , Peter B. Danzig , Michael N. Nelson
Proceedings of the second international conference on Architectual support for programming languages and operating systems October 1987
Volume 15 , 22 , 21 Issue 5 , 10 , 4

2 RISC versus CISC: a tale of two chips 90%
Dileep Bhandarkar
ACM SIGARCH Computer Architecture News March 1997
Volume 25 Issue 1
This paper compares an aggressive RISC and CISC implementation built with comparable technology. The two chips are the Alpha 21164 and the Intel Pentium® Pro processor. The paper presents performance comparisons for industry standard benchmarks and uses performance counter statistics to compare various aspects of both designs.

3 An architectural framework for migration from CISC to higher performance platforms 89%
Gabriel M. Silberman , Kemal Ebcioğlu
Proceedings of the 6th international conference on Supercomputing August 1992
We describe a novel architectural framework that allows software applications written for a given Complex Instruction Set Computer (CISC) to migrate to a different, higher performance architecture, without a significant investment on the part of the application user or developer. The framework provides a hardware mechanism for seamless switching between two instruction sets, resulting in a machine that enhances application performance while keeping the same program behavior (from a user per ...

4 The interaction of architecture and operating system design 87%
Thomas E. Anderson , Henry M. Levy , Brian N. Bershad , Edward D. Lazowska
Proceedings of the fourth international conference on Architectural support for programming languages and operating systems April 1991
Volume 26 , 19 , 25 Issue 4 , 2 , Special Issue

5 Porting OpenVMS from VAX to Alpha AXP 85%
Nancy Kronenberg , Thomas R. Benson , Wayne M. Cardoza , Ravindran Jagannathan , Benjamin J. Thomas
Communications of the ACM February 1993
Volume 36 Issue 2

6 Conforming inverted data store for low power memory 85%
You-Sung Chang , Bong-Il Park , Chong-Min Kyung
Proceedings of the 1999 international symposium on Low power electronics and design August 1999

7 Binary translation 84%
Richard L. Sites , Anton Chernoff , Matthew B. Kirk , Maurice P. Marks , Scott G. Robinson
Communications of the ACM February 1993
Volume 36 Issue 2

8 An out-of-order execution technique for runtime binary translators 82%
Bich C. Le
Proceedings of the eighth international conference on Architectural support for programming languages and operating systems October 1998
Volume 32 , 33 Issue 5 , 11
A dynamic translator emulates an instruction set architccturc by translating source instructions to native code during execution. On statically-scheduled hardware, higher performance can potentially be achieved by reordering the translated instructions; however, this is a challenging transformation if the source architecture supports precise exception semantics, and the user-level program is allowed to register exception handlers. This paper presents a software technique which allows a translato ...

9 Some correctness principles for machine language programs and microprograms 82%
W. D. Maurer
Conference record of the 7th annual workshop on Microprogramming September 1974
A machine-language program, or a microprogram implemented in writable control store, may modify itself. In order to prove the correctness of such a program, we must take this into account. Even if the program does not modify itself, we must prove this. Sometimes this may be done by looking at the individual instructions of the program; sometimes it must be tied in with the proof of correctness of the program. We state here, and illustrate by examples, certain principles for prov ...

10 Usenet Nuggets 82%
ACM SIGARCH Computer Architecture News March 1991
Volume 19 Issue 1

11 Characterization of alpha AXP performance using TP and SPEC workloads 80%
Z. Cvetanovic , D. Bhandarkar
ACM SIGARCH Computer Architecture News , Proceedings of the 21ST annual international symposium on Computer architecture April 1994
Volume 22 Issue 2
The characteristics of several commercial and technical workloads on the DEC 7000 AXP system are compared using built-in hardware monitors. The data analyzed include total instructions, cycles, multiple-issued instructions, stall components, cache misses, and instruction types. The data indicates that the two classes of workloads have vastly different characteristics and impose different requirements on the system design. Compared to VAX, Alpha AXP takes advantage of lower cycles per instruction ...

12 MOVE: a framework for high-performance processor design 80%
Henk Corporaal , Hans (J.M.) Mulder
Proceedings of the 1991 ACM/IEEE conference on Supercomputing August 1991

13 Considerations for new tactical computer systems 80%

Jon C. Strauss , Kenneth J. Thurber
ACM SIGARCH Computer Architecture News , Proceedings of the 4th annual symposium on Computer architecture March 1977
Volume 5 Issue 7

The real-time command and control environments characteristic of tactical military systems and industrial process control systems place unique and conflicting design requirements on a support computer system. These requirements include fast context switching, selective protection of programs and their files, controlled sharing of program and files, high processing speed, flexible, yet fast priority structure for interrupts and program execution, flexible high-speed I/O and flexible intercom ...

14 Programmed word length computer 80%

A. L. Lucke
Proceedings of the 1967 22nd national conference January 1967

The concept of a programmable word length computer has evolved through an attempt to minimize wasted storage in any fixed word length computer. Whether the computer contains six bits per word or 72 bits per word, storage is inevitably wasted when it is necessary to use a full word for a simple on-off switch, or to have 12-digit capacity used for 2-digit significance. Programmers have long recognized this. Thus, word packing and unpacking soon becomes a part of every programmer's repertoire. ...

15 Run-time checking in Lisp by integrating memory addressing and range checking 80%

M. Sato , S. Ichikawa , E. Goto
ACM SIGARCH Computer Architecture News , Proceedings of the 16th annual international symposium on Computer architecture April 1989
Volume 17 Issue 3

This paper describes the BL addressing mode and the address tag in FLATS2 machine, which is a general-purpose MIMD computer now under construction. The BL addressing mode integrates memory accessing and range checking by hardware. Address tag is a bit in word, which indicates the capability for memory access. Combining them together, efficient memory protection is provided at run-time. It reduces the cost of run-time type checking in Lisp by checking the address tag and the address of a poi ...

16 The design and development of a dynamic program behavior measurement tool for the Intel 8086/88 80%

R. J. Schwartz
ACM SIGARCH Computer Architecture News June 1989
Volume 17 Issue 4

When modeling a computer system, it is necessary to study the system's dynamic behavior. Examples of this behavior are branching frequency and operating system usage patterns. The dynamic properties of the system characterize its performance. Models make assumptions about such behavior, but require genuine data to validate the assumptions. In this article, we present a measurement tool that will collect and analyze dynamic program information for the Intel 8086/88. The design and development of ...

17 Value speculation scheduling for high performance processors 80%

Chao-Ying Fu , Matthew D. Jennings , Sergei Y. Larin , Thomas M. Conte
Proceedings of the eighth international conference on Architectural support for programming languages and operating systems October 1998
Volume 32 , 33 Issue 5 , 11

Recent research in value prediction shows a surprising amount of predictability for the values produced by register-writing instructions. Several hardware based value predictor designs have been proposed to exploit this predictability by eliminating flow dependencies for highly predictable values. This paper proposed a hardware and software based scheme for value speculation scheduling (VSS). Static VLIW scheduling techniques are used to speculate value dependent instructions by scheduling them ...

18 Instruction prefetching of systems codes with layout optimized for reduced cache misses 80%
Chun Xia , Josep Torrellas
ACM SIGARCH Computer Architecture News , Proceedings of the 23rd annual international symposium on Computer architecture May 1996
Volume 24 Issue 2
High-performing on-chip instruction caches are crucial to keep fast processors busy. Unfortunately, while on-chip caches are usually successful at intercepting instruction fetches in loop-intensive engineering codes, they are less able to do so in large systems codes. To improve the performance of the latter codes, the compiler can be used to lay out the code in memory for reduced cache conflicts. Interestingly, such an operation leaves the code in a state that can be exploited by a new type of ...

19 A fast and flexible performance simulator for micro-architecture trade-off analysis on UltraSPARC-I 80%
Marc Tremblay , Guillermo Maturana , Atsushi Inoue , Les Kohn
Proceedings of the 32nd ACM/IEEE conference on Design automation conference January 1995

20 Hardware and software support for efficient exception handling 80%
Chandramohan A. Thekkath , Henry M. Levy
Proceedings of the sixth international conference on Architectural support for programming languages and operating systems November 1994
Volume 29 , 28 Issue 11 , 5
Program-synchronous exceptions, for example, breakpoints, watchpoints, illegal opcodes, and memory access violations, provide information about exceptional conditions, interrupting the program and vectoring to an operating system handler. Over the last decade, however, programs and run-time systems have increasingly employed these mechanisms as a performance optimization to detect normal and expected conditions. Unfortunately, current archi ...

Results 1 - 20 of 196 **short listing**

Prev Page 1 2 3 4 5 6 7 8 9 10 Next Page

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.